Mr Long Summary ON ADVANCED ARRAYS

Finding the correct position in a sorted array

Before you can insert into an array, if the array is sorted AND you are just given the value, you need to determine the *CORRECT POSITION* for the new value.

Sample Code – Find Correct Position

```
sInput := InputBox( ‘Insert’, ’What value must be inserted into the array?’, ‘’ );
          // sInput is what must be inserted
bFound := false ;          //changes to true when position found
iLoop := 1 ;               //looping variable

while (bFound = false) AND ( iLoop <= ArraySize ) do        //while we haven’t found the position
 begin                                                      //AND we not at the end of the array
   if Array[ iLoop ] >= sInput then    //when found the right position
    begin                            //This array is sorted ASC, if DESC then > changes to <
      bFound := true ;
      iPos := iLoop ;      //record correct position for insertion
   end ; //end of if
inc(iLoop) //increase looping variable
 end; //end of while

if bFound = false then                    // if you didn’t find position to insert into Array
  Array[ ArraySize + 1 ] := sInput ;      // then you insert it at the end
else
 begin
  //do Insertation Code here <see next concept>
 end;  //end of else
```

Inserting into an array

If you are given the position to insert, then use the code below. If you are given just the value and you must determine the position in a sorted array, first use the algorithm above.

Sample Code – Inserting into an array

```
sInput := InputBox( 'Insert', 'What value must be inserted into the array?', '' );
        // sInput is what must be inserted
iPos := StrToInt( InputBox( 'Insert', 'What position in the array must it be inserted?', '' ) );
        // iPos is what position must be inserted

for K := ArraySize downto iPos do     //loop from the back until the correct position
 Begin
  Array[ K + 1 ] := Array[ K ]
  End;           //this moves elements up one position starting from the back till point of insertion

Array[ iPos ] := sInput ;        //store new name in required position
Inc( ArraySize )                 //increase Array size because new element has been inserted
```

Removing from an array

If you are given the position to remove, then use the code below. If you are given just the value and you must use a search algorithm (Grade 11) to find the position, then use the algorithm below.

Sample Code – Removing from an array

```
iPos := StrToInt( InputBox( 'Remove', 'What position in the array must it be removed?', '' ) );
        // iPos is what position must be removed

for K := iPos to ArraySize - 1 do     //loop from the correct position until 1 before the end
 Begin
  Array[ K ] := Array[ K + 1 ]
  End;           //this moves elements down one position starting from point of insertion till end
Dec( ArraySize )         //decrease Array size because element has been removed
```

Remove duplicates from an array

Declare another array (***arrNoDup***) with an *ArraySize* variable for that array (***iNoDupSize***).

Sample Code – Removing Duplicates

```
iNoDupSize := 0 ;                   //no values in your arrNoDup to start with
for K := 1 to ArraySize - 1 do      //loop from 1 to 1 before end of array
 Begin
  iLoop := K + 1 ;          // start looping variable from one after value you are checking till end
  bDup := false ;           // becomes true when a duplicate is found
  while ( iLoop <= ArraySize ) AND ( bDup = false ) do     //while we haven't reached end of array
   begin                                                   //AND we haven't found a duplicate
    if Array[ K ] = Array[ iLoop ] then          // if duplicate found
     bDup := true ;
    inc( iLoop ) ;          // increase looping variable
   end; // end of while

   If bDup = false then             //no duplicate was found after all that checking
   Begin                            //then we have found the last / only version of that value
     Inc( iNoDupSize ) ;
     arrNoDup [ iNoDupSize ] := Array[ K ] ;  // add element to arrWithDup array
   end;  // end of if
 End;  //end of for loop

Inc( iNoDupSize ) ;         // last Element can't be compared with any value and must therefore also
arrNoDup [ iNoDupSize ] := Array[ ArraySize ] ;        // be assigned to arrNoDup
```

Merge TWO arrays

First add all the elements that appear in both arrays (***arrOne*** and ***arrTwo***) into one array ***arrBoth***. If you want unique values from both arrays *(if a value appears in both arrays, you only want one version of that value)* then use the *Remove Duplicates* algorithm above on the combined array (***arrBoth***)

Sample Code – Merge two arrays

```
iArraySizeBoth := 0 ;
For K := 1 to iArraySizeOne  do          //loop through arrOne
 Begin                                   //add all elements from arrOne to arrBoth
   Inc( iArraySizeBoth ) ;
   arrBoth[ iArraySizeBoth ] := arrOne[ K ] ;
 End ;  //end of For

For K := 1 to iArraySizeTwo  do          //loop through arrTwo
 Begin                                   //add all elements from arrTwo to arrBoth
   Inc( iArraySizeBoth ) ;
   arrBoth[ iArraySizeBoth ] := arrTwo[ K ] ;
 End ;  //end of For
//now arrBoth contains both numbers from arrOne and arrTwo
//run Remove Duplicates algorithm to get only unique numbers in the merged array
```

Intersection of TWO arrays

One array (***arrBoth***) contains all the elements that appear in both arrays (***arrOne*** and ***arrTwo***).

Sample Code – Intersection of two arrays

```
iArraySizeBoth := 0 ;
For K := 1 to iArraySizeOne  do          //loop through arrOne
  Begin
    iLoop := 1 ;                //looping variable to loop through arrTwo
    bFound := false ;           //becomes true when we find element in both arrays
    while ( iLoop <= iArraySizeTwo ) AND ( bFound = false ) do      //while not at end of arrTwo
      begin                                                         //AND no match found
        If arrOne[ K ] = arrTwo[ iLoop ] then //element in arrOne is same is element in arrTwo
         Begin
          bFound := true ;
          inc( iArraySizeBoth ) ;                   //add element to arrBoth array
          arrBoth[ iArraySizeBoth ] := arrOne[ K ]  ;
         End ;   //end of if
        Inc( iLoop) ;    //increase arrTwo looping variable
      End; //end of while
  End; //end of for
```

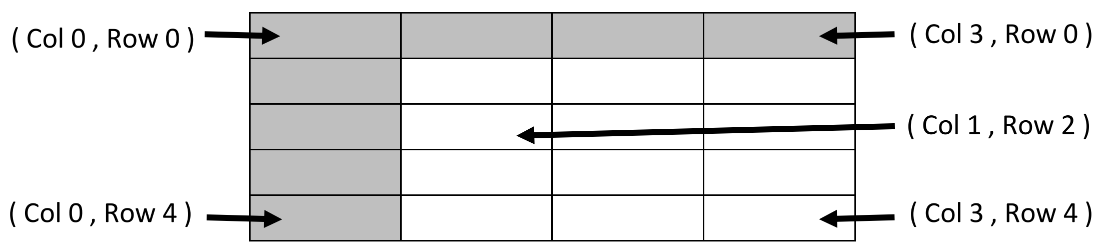
String grids

- Component found under the Additional component tab.
- Used to display data in a table format.
- Properties to take note of
    - *RowCount* and *ColCount* determine the number of rows and columns respectively.
    - *FixedCols* and *FixedRows* determine the number shaded rows and columns respectively. This are often used for headings.
    - *DefaultColWidth* and *DefaultRowWidth* determine the width of rows and columns respectively.
    - *Cells* property needs a column integer position and row integer position *<in that order – just remember alphabetical **c**ol then **r**ow>* and that block stores a **string** value.

      Example: sgdData.Cells[iCol , iRow] := 'Hello' ;

Position in a string grid

- Each column starts at position 0 and the last column is the *ColCount* – 1.
- Each row starts at position 0 and the last row is the *RowCount* – 1.
- *HINT: If you are using a fixed column and row, then the values in those cells will be in column 0 (row 1 to end), for the headings in the left side and in row 0 (column 1 to end), for the headings in the top row.*
- Example:



- Example of enter data into a string grid:

```
stringGrid1.Cells[ 0 , 0 ] := 'Hello' ;
stringGrid1.Cells[ 2 , 4 ] := 'X' ;
stringGrid1.Cells[ 1 , 1 ] := '23' ;
for K := 1 to 4 do
 stringGrid1,Cells[ 0 , K ] := IntToStr( K ) ;
```

Hello			
1	**23**		
2			
3			
4		**X**	

2D Arrays

- Array that stores data in a table format.
- Declaration examples:
  arrOne : array[1..4 , 1..6] of integer ; **//4 x 6 table where each element is an integer**
  arrTwo : array['A'..'C' , 1..5] of real ; **//3 x 5 table where each element is a real**
  arrThree : array[1..5 , 1..5] of string ; **//5 x 5 table where each element is a string**

- Read values into a 2D array examples:
  arrOne[1 , 1] := 34 ;
  arrTwo['B' , 3] := 23.5 ;
  for K := 1 to 5 do *arrThree*[K , 1] := 'X' ; ***//each element in first row will contain X***

- When accessing all the values in a 2D array, you will need a nested for loop.

Example algorithms when using a 2D array of numerical values

- Find average of all the numbers in a numerical 2D array
  NOTE: This also includes finding the sum of all the numbers.

Sample Code – Average ALL in 2D array

```
//Using arrTemp : array[ 1..5 , 1..10 ] of integer ;

iSum := 0 ;
For J := 1 to 10  do      //loop through columns
 For K := 1 to 5  do      //loop through rows
  Begin
    iSum := iSum + arrTemp[ K , J ] ;  //add element in array to Sum variable
  end ; //end of K for loop

rAve := iSum / ( J * K ) ;         //J * K = number of elements in array <If it is full of values>
```

- Find minimum value of all the numbers in a numerical 2D array

Sample Code – Minimum of ALL in 2D array

```
//Using arrTemp : array[ 1..5 , 1..10 ] of integer ;

iMin := 9999 ;            //initialise iMin to the opposite of what you want, a very large number
For J := 1 to 10  do              //loop through columns
 For K := 1 to 5  do              //loop through rows
  Begin
    if arrTemp[ K , J ] < iMin then   //found a new minimum value
        Begin
        iMin := arrTemp[ K , J ] ;        //record the new minimum value
        iRowPos := K ;
        iColPos   := J  ;           //record new minimum value's position in 2D array
        end ; //end of if
   end ; //end of K for loop
 //For maximum value, use iMax variable and initialise iMax to really small number iMax := -9999
//and change sign in if statement from < to >
```

NOTE: In the above examples, it doesn't matter which order you loop the rows and columns.

- Find the sum of EACH ROW or EACH COLUMN in a numerical 2D array
  In this case, the order of for loops is important.
  - For each row, you loop by ROW first, then by column
  - For each column, you loop by COLUMN first, then by row

Sample Code – Sum of EACH ROW in 2D array

```
//Using arrTemp : array[ 1..5 , 1..10 ] of integer ;

For K := 1 to 5  do       //loop through rows
Begin
iSum := 0 ;        //initialise Sum inside the first loop now because you
                   //need to reset it before you Sum the next row
 For J := 1 to 10  do    //loop through columns
  Begin
    iSum := iSum + arrTemp[ K , J ] ;  //add element in array to Sum variable
  end ; //end of J for loop
 showmessage('Sum of Row ' + IntToStr( K ) + ' = ' + IntToStr( iSum ) ;
  //Display answer once Sum is complete for EACH row
 end ; //end of K for loop

//We have 5 answers now so we display them individually in a showmessage or in a memo
//control or you can extend the ColCount of a string grid and place answer at the end of each row
// in last column <see display in string grid later>
```

Sample Code – Sum of EACH COL in 2D array

```
//Using arrTemp : array[ 1..5 , 1..10 ] of integer ;

For J := 1 to 10  do      //loop through columns
Begin
iSum := 0 ;        //initialise Sum inside the first loop now to reset it before you Sum the next col
 For K := 1 to 5  do     //loop through rows
  Begin
    iSum := iSum + arrTemp[ K , J ] ;  //add element in array to Sum variable
  end ; //end of J for loop
 showmessage('Sum of Column ' + IntToStr( J ) + ' = ' + IntToStr( iSum ) ;
  //Display answer once Sum is complete for EACH row
 end ; //end of K for loop
```

Displaying a 2D array in a string grid

- If there are no fixed rows or columns, then the string grid first row and column starts at 0

Sample Code – Display 2D array in string grid

```
//Using arrTemp : array[ 1..5 , 1..10 ] of integer ;
// stringGrid1.RowCount := 5 ;
// stringGrid1.ColCount  := 10 ;
For J := 1 to 10  do      //loop through columns
 For K := 1 to 5  do      //loop through rows
  Begin
    stringGrid1.Cells[ J - 1 , K - 1 ] := IntToStr( arrTemp[ K , J ] ) ;
   //because stringGrid goes from 0 – 9 for rows and 0 – 4 for columns that's why you need the -1
  end ; //end of K for loop
```

- If there are is 1 fixed row and column added, then the following line can replace the one in the code segment above:
  stringGrid.Cells[J , K] := IntToStr(arrTemp[K , J]) ;

Additional Links:

- Youtube video playlist:
  https://www.youtube.com/watch?v=VGRRpT-1CQM&list=PLxAS51iVMjv9-EGk4_-ktl33SQC716z8g
- Google drive resource activities:
  https://tinyurl.com/MLE-G12IT-AdvancedArrays